IN THE CLAIMS:

Please amend Claims 2, 4, 6, 8 and 20 as follows:

2. (Amended) The method as claimed in claim 1, further comprising the steps of:

using a first bit stream of an IP address of an incoming IP packet as an index to look up said segmentation table;

determining if said first bit stream of an entry of said segmentation table pointed to by said index is larger than or equal to M;

outputting said entry of said segmentation table as a next hop when said first bit stream of said entry is smaller than said M.

determining a correspondent entry of said segmentation table is a pointer pointing to a corresponding Next Hop Array when said first bit stream of said entry of said segmentation table is larger than or equal to said M, and a second bit stream of said corresponding entry of said segmentation table is smaller than or equal to Y, and using said second bit stream of said corresponding entry of said segmentation table as an index to look up said corresponding Next Hop Array.

4. (Amended) The method as claimed in claim 1, further comprising the steps of:

determining a correspondent entry of said segmentation table is a pointer pointing to a corresponding Code Word Array when a first bit stream of

said entry is larger than or equal to M, and a second bit stream of said correspondent entry of said segmentation table is larger than said Y;

computing an index for looking up a corresponding code word in said Code Word Array by adding said correspondent entry of said segmentation table, which is a pointer, plus said second bit stream;

computing an index for looking up a corresponding Compressed Next Hop Array by adding (said pointer + $2^{k-4}\times4 - 1$), a Base of said corresponding Code Word and $|w|$, said $|w|$ representing the number of "1"s accumulated from the *0-th* bit to *w-th* bit of the map of the code word of said corresponding Code Word Array, and said *k* representing an offset length.

6. (Amended) The method as claimed in claim 1, wherein each of said Next Hop Array contains 2^k entries, and said k is determined by the longest prefix length of each segment.

8. (Amended) The method as claimed in claim 7 comprises the steps of:

reading a set of sorted route prefixes of a segment in an increasing order by the length of prefixes, and each pair of start point and end point of said list of sorted route prefixes is sorted according to the order in said set of route prefixes of said segment ;

sorting each element in said set in an order according to its memory address in said segment;

processing each element in said set from left to right and in a manner that:

(a) determining if a selected element is a start point S_i^0, where "i" represents the *i-th* route prefix in said set and "0" represents the number of update times of said start point in the memory; when said selected element is a start point S_i^0, executing step (b), and when said selected element is not a start point S_i^0, executing step (c);

(b) pushing said start point S_i^0 onto a stack, and appending said start point S_i^0 to an array; repeate said step (a) until each element in said set is processed;

(c) removing a top element of said stack;

(d) determining if the top element of said stack is a start point S_j^k, where "*j*" represents the *j-th* route prefix in said set and "k" represents the number of update times of said start point in the memory; when said top element is said start point S_j^k, executing step (e), and when said top element is not said start point S_j^k, executing step (f);

(e) appending S_j^{k+1} to said array where the next hop of a start point S_j^{k+1} is equal to the next hop of a start point S_j^k, and the memory address of a start point S_j^{k+1} is equal to the memory address of an end address E_i^0+1; and replacing the top element of said stack with said start point S_j^{k+1}; repeate said step (a) until each element in said set is processed;

(f) executing nothing; repeate said setp (a) until each element in said set is processed;